Как (не) выстрелить себе в ногу на Lua

Ярослав Дынников





Lua очень простой

```lua
local function hello(name)
    print('Hello, ' .. name .. '!')
end

local names = {'World', 'Highload', 'Moscow'}

for i, name in pairs(name) do
    hello(name)
end
```

```
Hello, World!
Hello, Moscow!
Hello, Highload!
```




Типизация

- Динамическая
- Не сильная (но почти)

Пример из жизни

```
for uuid, new_leader in pairs(new_leaders) do

    log.info('Replicaset %s: new leader %s, was %s',
        uuid,
        describe(new_leader),
        describe(old_leaders[uuid])
    )

end
```

Пример из жизни

```
for uuid, new_leader in pairs(new_leaders) do

+   if uuid == my_uuid then
+       uuid = uuid .. ' (me)'
+   end

    log.info('Replicaset %s: new leader %s, was %s',
        uuid,
        describe(new_leader),
        describe(old_leaders[uuid])
    )

end
```

Пример из жизни

```
for uuid, new_leader in pairs(new_leaders) do

+   if uuid == my_uuid then
+       uuid = uuid .. ' (me)'
+   end

    log.info('Replicaset %s: new leader %s, was %s',
        uuid,
        describe(new_leader),
        describe(old_leaders[uuid]) -- nil (иногда)
    )

end
```

Пример из жизни

```
for uuid, new_leader in pairs(new_leaders) do

+   local replicaset_name = uuid

    if uuid == my_uuid then
-       uuid = uuid .. ' (me)'
+       replicaset_name = replicaset_name .. ' (me)'
    end

    log.info('Replicaset %s: new leader %s, was %s',
-       uuid,
+       replicaset_name,
        describe(new_leader),
        describe(old_leaders[uuid])
    )

end
```

Типизация — не сильная

Нельзя

```
"k" .. nil -- attempt to concatenate a nil value
{1, 2} + {3} -- attempt to perform arithmetic on a table value
-- Python: [1, 2] + [3] == [1, 2, 3]
-- JS:     [1, 2] + [3] == '1,23'
```

Типизация — не сильная

Нельзя

```
"k" .. nil -- attempt to concatenate a nil value
{1, 2} + {3} -- attempt to perform arithmetic on a table value
-- Python: [1, 2] + [3] == [1, 2, 3]
-- JS:     [1, 2] + [3] == '1,23'
```

Можно

```
math.sqrt("144") -- 12 (number)

string.len(1337) -- 4
```

Lua 5.1 Reference Manual. §2.2.1 Coercion



Аргументы надо проверять

```
function get_stat(uri, opts)
    return http.get('http://' .. uri .. '/stat', opts)
end
```




Аргументы надо проверять

```lua
function get_stat(uri, opts)
    return http.get('http://' .. uri .. '/stat', opts)
end
```

```lua
get_stat(req.uri) -- req.uri == nil
-- error: api.lua:310: attempt to concatenate a nil value
-- Не понятно
```




Аргументы надо проверять

```
function get_stat(uri, opts)
    assert(type(uri) == 'string', 'uri must be a string')
    -- ...
end
```




Аргументы надо проверять

```lua
function get_stat(uri, opts)
    assert(type(uri) == 'string', 'uri must be a string')
    -- ...
end
```

```lua
get_stat(req.uri) -- req.uri == nil
-- error: api.lua:310: uri must be a string
-- Уже лучше
```




Аргументы надо проверять

```lua
function get_stat(uri, opts)
    assert(type(uri) == 'string', 'uri must be a string')
    -- ...
end
```

```lua
get_stat(req.uri) -- req.uri == nil
-- error: api.lua:310: uri must be a string
-- Уже лучше
```

```lua
get_stat('localhost', {timeuot = 1})
--                        ^^ typo
-- Ошибки нет, но поведение не правильное
```




Аргументы надо проверять

```lua
require('checks')

function get_stat(uri, opts)
    checks('string', {timeout = '?number'})
    -- ...
end
```




Аргументы надо проверять

```
require('checks')

function get_stat(uri, opts)
    checks('string', {timeout = '?number'})
    -- ...
end
```

```
get_stat()
-- error: bad argument #1 to get_stat (string expected, got nil)
```




Аргументы надо проверять

```lua
require('checks')

function get_stat(uri, opts)
    checks('string', {timeout = '?number'})
    -- ...
end
```

```lua
get_stat()
-- error: bad argument #1 to get_stat (string expected, got nil)
```

```lua
get_stat('localhost', {timeuot = 1})
-- error: unexpected argument opts.timeuot to get_stat
```




Типизация — простая, как топор

- boolean, string, number
- function, thread
- userdata, cdata
- table
- nil



number == double



number == double

```
assert_equals(0.1 + 0.2, 0.3)
-- error:
-- expected: 0.3
--   actual: 0.3
```

number == double

```
assert_equals(0.1 + 0.2, 0.3)
-- error:
-- expected: 0.3
--   actual: 0.3
```

```
2^53 - 2 -- 9007199254740990
2^53 - 1 -- 9007199254740991
2^53 + 0 -- 9007199254740992
2^53 + 1 -- 9007199254740992
2^53 + 2 -- 9007199254740994

2^53 + 1 == 2^53 -- true
```

number == double

```
assert_equals(0.1 + 0.2, 0.3)
-- error:
-- expected: 0.3
--   actual: 0.3
```

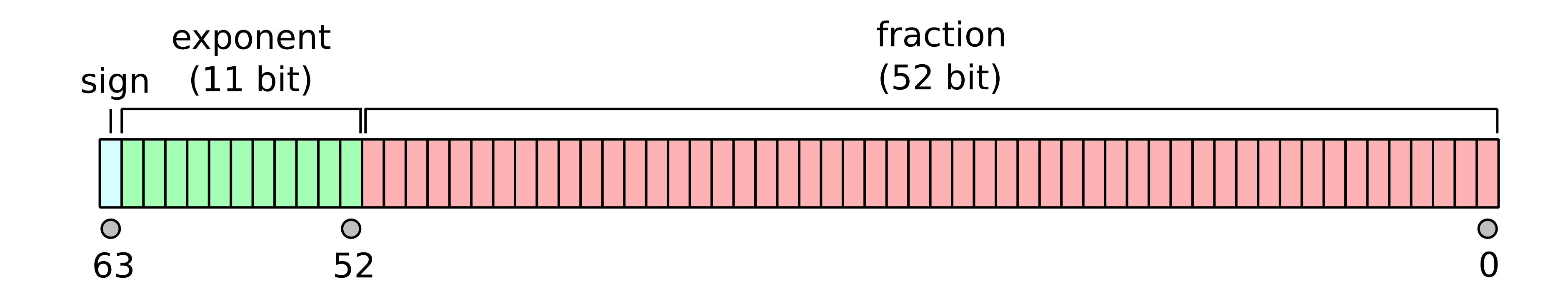
```
2^53 - 2 -- 9007199254740990
2^53 - 1 -- 9007199254740991
2^53 + 0 -- 9007199254740992
2^53 + 1 -- 9007199254740992
2^53 + 2 -- 9007199254740994

2^53 + 1 == 2^53 -- true
```

```
now = clock.time64() -- 1621068872741010434, ~2^60
ffi.typeof(t) -- ctype<uint64_t>
```



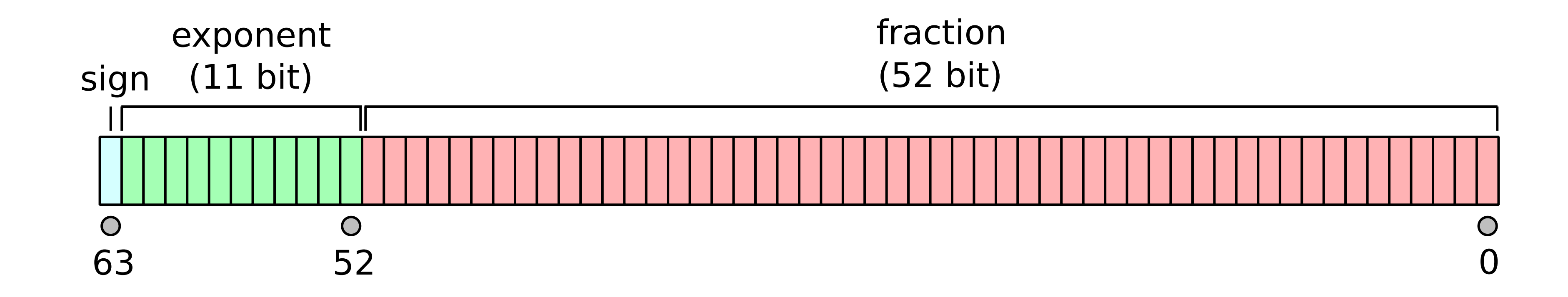

IEEE 754



$$(-1)^{sign} \cdot 2^{(e-1023)} \cdot \left(1 + f \cdot 2^{-52}\right), e \in (0, 2047)$$



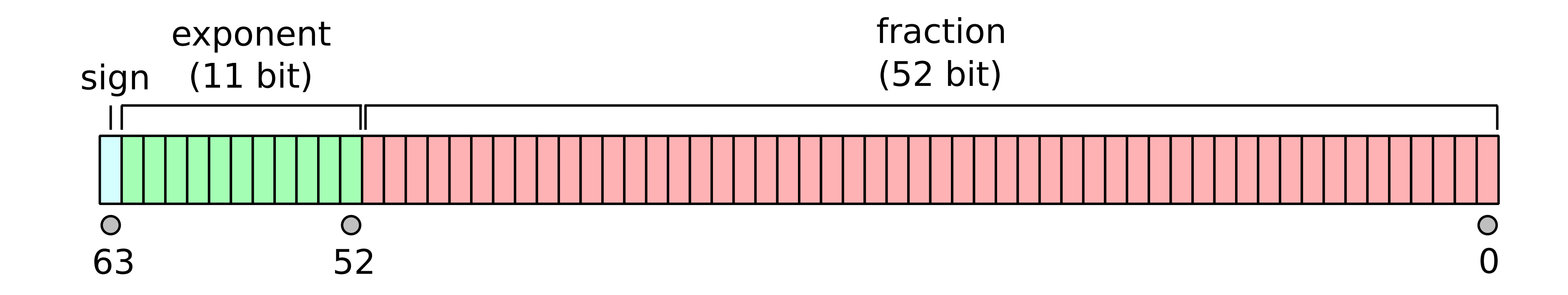
IEEE 754



$$(-1)^{sign} \cdot 2^{(e-1023)} \cdot \left(1 + f \cdot 2^{-52}\right), e \in (0, 2047)$$
$$(-1)^{sign} \cdot 2^{(e-1023)} \cdot \left(0 + f \cdot 2^{-52}\right), e = 0$$



IEEE 754



$$(-1)^{sign} \cdot 2^{(e-1023)} \cdot \left(1 + f \cdot 2^{-52}\right), e \in (0, 2047)$$
$$(-1)^{sign} \cdot 2^{(e-1023)} \cdot \left(0 + f \cdot 2^{-52}\right), e = 0$$

```
0 11111111111 0000000000000000000000000000000000000000000000000000₂ == +inf
1 11111111111 0000000000000000000000000000000000000000000000000000₂ == -inf
s 11111111111 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx₂ ==  nan
```




Not a Number

```
nan = math.sqrt(-1)
nan = math.huge / math.huge
nan = 0 / 0
```

Not a Number

```
nan = math.sqrt(-1)
nan = math.huge / math.huge
nan = 0 / 0
```

```
nan > nan -- false
nan < nan -- false
nan == nan -- false

nan ~= nan -- true

nan + 1 -- nan
nan * 2 -- nan
```




Not a Number

```
function assert_ge(l, r)
    if l < r then
        error("Assertion failed!")
    end
end
```




Not a Number

```lua
function assert_ge(l, r)
    if l < r then
        error("Assertion failed!")
    end
end
```

```diff
function assert_ge(l, r)
-    if l < r then
+    if not (l >= r) then
        error("Assertion failed!")
    end
end
```

Not a Number

```
1 ^ nan -- 1
nan ^ 0 -- 1
```




Not a Number

```
1 ^ nan -- 1
nan ^ 0 -- 1
```

```
$ man pow

SYNOPSIS
    #include <math.h>
    double pow(double x, double y);

RETURN VALUE

    If x is +1, the result is 1.0 (even if y is a NaN).
    If y is 0, the result is 1.0 (even if x is a NaN).
```

Linux man page



LuaJIT internal tags

```
// lj_obj.h

// Interpreted as a double these are special NaNs. The FPU only generates
// one type of NaN (0xfff8_0000_0000_0000). So MSWs > 0xfff80000 are available
// for use as internal tags.
//                   ---MSW---.---LSW---
// primitive types |  itype  |         |
// lightuserdata   |  itype  |  void * |
// GC objects      |  itype  |  GCRef  |
// int (LJ_DUALNUM)|  itype  |   int   |
// number           -------double------

#define LJ_TNIL     (~0u)
#define LJ_TFALSE   (~1u)
#define LJ_TTRUE    (~2u)
#define LJ_TSTR     (~4u)
#define LJ_TTAB     (~11u)
```

Таблицы снаружи

```
t1 = { 'Sunday', 'Monday', 'Im tired' }
```




Таблицы снаружи

```
t1 = { 'Sunday', 'Monday', 'Im tired' }
```

```
t2 = {
    cat = 'meow',
    dog = 'woof',
    cow = 'moo',
}
```




Таблицы снаружи

```
t1 = { 'Sunday', 'Monday', 'Im tired' }
```

```
t2 = {
    cat = 'meow',
    dog = 'woof',
    cow = 'moo',
}
```

```
t3 = {
    'k1', 'k2', 'k3',
    ['k1'] = 'v1',
    ['k2'] = 'v2',
    ['k3'] = 'v3',
}
```




Таблицы изнутри

```
typedef struct GCtab {
  /* GC stuff */
  MRef array;     /* Array part. */
  MRef node;      /* Hash part.  */
  uint32_t asize; /* Size of array part (keys [0, asize-1]). */
  uint32_t hmask; /* Hash part mask (size of hash part - 1). */
} GCtab;
```

Таблицы изнутри

```
typedef struct GCtab {
  /* GC stuff */
  MRef array;      /* Array part. */
  MRef node;       /* Hash part.  */
  uint32_t asize; /* Size of array part (keys [0, asize-1]). */
  uint32_t hmask; /* Hash part mask (size of hash part - 1). */
} GCtab;
```

Таблицы изнутри

```
typedef struct GCtab {
  /* GC stuff */
  MRef array;     /* Array part. */
  MRef node;      /* Hash part.  */
  uint32_t asize; /* Size of array part (keys [0, asize-1]). */
  uint32_t hmask; /* Hash part mask (size of hash part - 1). */
} GCtab;
```




Таблицы изнутри

```
t = {}
-- table: 0x40eae3a8
--   a[0]:
--   h[1]: nil=nil
```




Таблицы изнутри

```
t = {}
-- table: 0x40eae3a8
--  a[0]:
--  h[1]: nil=nil
```

```
t["a"] = "A"
t["b"] = "B"
t["c"] = "C"
-- table: 0x40eae3a8
--  a[0]:
--  h[4]: b=B, nil=nil, a=A, c=C
```




Таблицы изнутри

```
t1 = {a = 1, b = 2, c = 3}
-- table: 0x40eaeb08
--  a[0]:
--  h[4]: b=2, nil=nil, a=1, c=3

t2 = {c = 3, b = 2, a = 1}
-- table: 0x40ea7e70
--  a[0]:
--  h[4]: b=2, nil=nil, c=3, a=1
```

Таблицы изнутри

```
t1 = {a = 1, b = 2, c = 3}
-- table: 0x40eaeb08
--  a[0]:
--  h[4]: b=2, nil=nil, a=1, c=3

t2 = {c = 3, b = 2, a = 1}
-- table: 0x40ea7e70
--  a[0]:
--  h[4]: b=2, nil=nil, c=3, a=1
```

```
traverse(pairs, t1)
-- b=2, a=1, c=3

traverse(pairs, t2)
-- b=2, c=3, a=1
```




Таблицы изнутри

```
t2["c"] = nil
-- table: 0x411c83c0
--  a[0]:
--  h[4]: b=2, nil=nil, c=nil, a=1
```

Таблицы изнутри

```
t2["c"] = nil
-- table: 0x411c83c0
--  a[0]:
--  h[4]: b=2, nil=nil, c=nil, a=1
```

```
next(t2, "c") -- a
next(t2, "d") -- error: invalid key to 'next'
```




Sequence

```
t = {1, 2}
-- table: 0x41735918
--  a[3]: nil, 1, 2
--  h[1]: nil=nil
```

Sequence

```
t = {1, 2}
-- table: 0x41735918
--  a[3]: nil, 1, 2
--  h[1]: nil=nil
```

```
t = {[2] = 2, 1}
-- table: 0x416a3998
--  a[2]: nil, 1
--  h[2]: nil=nil, 2=2
```

Sequence

```
t = {1, 2}
-- table: 0x41735918
--  a[3]: nil, 1, 2
--  h[1]: nil=nil
```

```
t = {[2] = 2, 1}
-- table: 0x416a3998
--  a[2]: nil, 1
--  h[2]: nil=nil, 2=2
```

```
t = table.new(4, 4)
for i = 1, 8 do t[i] = i end
-- table: 0x412c6df0
--  a[5]: nil, 1, 2, 3, 4
--  h[4]: 7=7, 8=8, 5=5, 6=6
```

Длина массива — определение

```
#{1, 2, 3} -- 3
```

Lua 5.1 Reference Manual. §3.4.6 – The Length Operator



Длина массива — определение

```
#{1, 2, 3} -- 3
```

Undefined behavior:

```
#{nil, 2} -- 2
-- table: 0x410d5528
--  a[3]: nil, nil, 2
--  h[1]: nil=nil

#{[2] = 2} -- 0
-- table: 0x410d5810
--  a[0]:
--  h[2]: nil=nil, 2=2
```

Lua 5.1 Reference Manual. §3.4.6 – The Length Operator



Откуда берутся дырки?

```diff
- t[i] = nil
+ table.remove(t, i)
```




Откуда берутся дырки?

```
- t[i] = nil
+ table.remove(t, i)
```

```
function vararg(...)
    local args = {...}
    -- #args == undefined behavior
end

vararg(nil, "err")
```




Длина массива — применение

```
table.sort(t) -- 1, #t
unpack(t) -- 1, #t
```




Длина массива — применение

```
table.sort(t) -- 1, #t
unpack(t) -- 1, #t
```

```
function table.pack(...)
    return {n = select('#', ...), ...}
end

t = table.pack(nil, 2)
unpack(t, 1, t.n) -- nil, 2
```




Итерации

```lua
for i = 1, #t do
end

for i, v in ipairs(t) do
end
```




Итерации

```lua
for i = 1, #t do
end

for i, v in ipairs(t) do
end
```

```lua
-- ipairs:
local i = 1
while type(t[i]) ~= 'nil' do
    -- do something
    i = i + 1
end
```

FFI и cdata

```
ffi = require('ffi')
NULL = ffi.new('void*', nil)
```




FFI и cdata

```
ffi = require('ffi')
NULL = ffi.new('void*', nil)

type(nil) -- nil
type(NULL) -- cdata
ffi.typeof(box.NULL) -- ctype<void *>
```

FFI и cdata

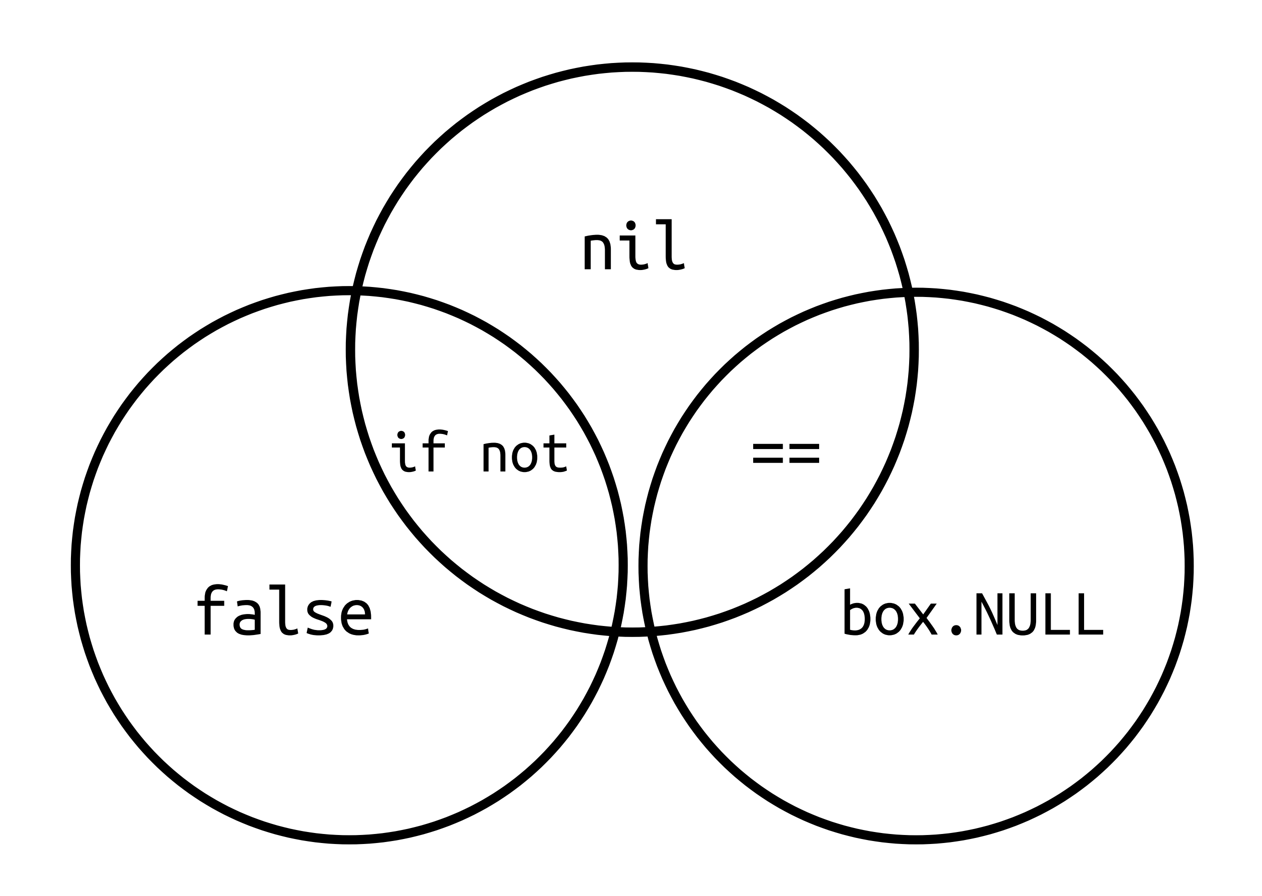
```
ffi = require('ffi')
NULL = ffi.new('void*', nil)

type(nil) -- nil
type(NULL) -- cdata
ffi.typeof(box.NULL) -- ctype<void *>

NULL == nil -- true

if NULL then
    print('NULL is not nil')
end
-- NULL is not nil
```

FFI и cdata





Выводы

- Старайтесь писать код без багов 😊
- Проверяйте аргументы, пользуйтесь линтерами
- Избегайте NaN
- Не делайте лишних предположений
- Бойтесь дырявых массивов

